SOURCEFORGE

IKVM.NET

Brought to you by: jfrijters

Ikvmc

Authors: Anonymous

{{UserGuide}}

The ikvmc tool converts Java bytecode to .NET dll's and exe's.

Usage

```
ikvmc [ options ] classOrJarfile [ classOrJarfile ... ]
```

options

```
See below.
```

classOrJarfile

```
Name of a Java .class or .jar file. May contain wildcards (*.class).
```

Options

OptionDescription

@<filename>
Read more options from file

-out:<outputfile>
pecifies the name of the output file. Should have a .dll extension (if -target is library) or an .exe extension (if -target is exe or winexe). In most cases, if you omit this option, ikvmc will choose an output name based on the -target and the name of the input files. However, if you specify input files using wildcards, you must use this option to specify the output file.

-assembly:<assembly-name>
Specifies the name of the generated assembly. If omitted, the assembly name is (usually) the output filename.

-target:target-type
Specifies whether to generate an .exe or .dll. target-type is one of

- exe - generates an executable that runs in a Windows command window
- winexe - generates an .exe for GUI applications
- library - generates a .dll
- module - generates a .netmodule

On Linux, there is no difference between exe and winexe.

-platform:<platform>
Limit which platforms this code can run on:

- x86 - 32 bit

- Itanium
- x64
- anycpu

The default is anycpu. This can be helpful if your Java code use a dll via JNI. For example a 32 bit dll should also run on a 64 bit system as 32 bit application.

-keyfile:<keyfilename>
Uses keyfilename to sign the resulting assembly.

-key<keycontainer>
Use keycontainer to sign the assembly

-version:<M.m.b.r>
Specifies the assembly version.

-fileversion:<version>
File version

-main:<classname>
Specifies the name of the class containing the main method. If omitted and the -target is exe or winexe, ikvmc searches for a qualifying main method and reports if it finds one.

-reference:<library-filespec>
If your Java code uses .NET API's, specify the dll's using this option. This option can appear more than once if more than one library is referenced. Wildcards are permitted (e.g. c:\libs\*.dll).

-recurse:<filespec>
Processes all files matching filespec in and under the directory specified by filespec. Example: -recurse:*.class

-nojni
Do not generate JNI stub for native methods

-resource:name=path
Includes path as a Java resource named name

-externalresource:<name>=<path>
Reference file as Java resource

-exclude:filename
filename is a file containing a list of classes to exclude

-debug
Generates debugging information in the output. Note that this is only helpful if the .class files contain debug information (compiled with the javac -g option).

-srcpath:path
Specifies the location of source code. Use with -debug. The package of the class is appended to the specified path to locate the source code for the class.

-Xtrace:name
Displays all tracepoints with name

-Xmethodtrace:methodname
Builds method trace into the specified output method.

Notes

The ikvmc tool generates .NET assemblies from Java class files and jar files. It converts the Java bytecodes in the input files to .NET CIL. Use it to produce

- .NET executables (-target:exe or -target:winexe)
- .NET libraries (-target:library)
- .NET modules (-target:module)

Java applications often consist of a collection of jar files. ikvmc can process several input jar files (and class files) and produce a single .NET executable or library. For example, an application consisting of main.jar, lib1.jar, and lib2.jar can be converted to a single main.exe.

When processing multiple input jar files that contain duplicate classes / resources, ikvmc will use the first class / resource it encounters, and ignore duplicates encountered in jars that appear later on the command line. It will produce a warning in this case. Thus, order of jar files can be significant.

Note

- When converting a Java application with ikvmc, for best results, list the jars on the ikvmc command line in the same order that they appear in the Java application's classpath.
  See also the concepts of classloader.
- [Ikvmc_messages]

Examples

Single jar file

```
ikvmc myProg.jar
```

Scans myprog.jar for a main method. If found, an .exe is produced; otherwise, a .dll is generated.

jar file and class files to a single exe

```
ikvmc -out:myapp.exe -main:org.anywhere.Main -recurse:bin\*.class lib\mylib.jar
```

Processes all .class files in and under the bin directory, and mylib.jar in the lib directory. Generates an executable named myapp.exe using the class org.anywhere.Main as the main method.

jar file to dll and class files to an exe

```
ikvmc mylib.jar
ikvmc -out:myapp.exe -main:org.anywhere.Main -recurse:bin\*.class -reference:mylib.dll
```

First it generate a library mylib.dll from mylib.jar. Then it generate executable named myapp.exe from all .class files in and under the bin directory using the class org.anywhere.Main as the main method. This make duplicate file names possible which are used from the Java service API.

Related

Wiki: ClassLoader
Wiki: Components
Wiki: Convert_a_jar_file_to_a_dll_and_use_it_as_library
Wiki: Ikvmc_messages
Wiki: Tools
Wiki: Tutorial